AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph beginning on page 8, line 22 as follows:

Equivalently, this object is achieved by the code generation method of ~~claim 12~~ the present invention. In particular, the object is achieved by the provision of the steps of (a) converting ~~said~~ the index k into ~~said~~ the modified index j, (b) initializing a counter value (code bit index) i (to zero, ~~e.g.~~ for example), (c) performing logic operations (only) on bits of ~~said~~ the modified index j and bits of ~~said~~ the counter value i, thereby generating a code bit of ~~said~~ the orthogonal code, (d) incrementing ~~said~~ the counter value i by one, and (e) repeating steps (c) and (d) until a desired number of code bits has been generated.

Please delete the paragraph on page 9, lines 26-28.

Please amend the paragraph beginning on page 9, line 29 as follows:

As the skilled person will readily appreciate, the ~~features of claims 1 and 12 do contribute to meeting these~~ present invention meets requirements (a) to (d) as described above with respect to the prior art independent from the type (OVSF, Hadamard, Walsh etc.) of orthogonal code to be generated (no matter whether fixed or selectable), and also independent from the particular realization of the index conversion and logic units (for the respective steps). In addition, the code generator of ~~claim 1~~ the present invention does not necessarily include a counter for generating the counter value i, as will be seen below.

Please amend the paragraph beginning on page 10, line 8 as follows:

~~According to claims 2 and 13, said~~ The corresponding code ~~is one of~~ may be an OVSF code, a Hadamard code, ~~and~~ or a Walsh code. In other words, the type of orthogonal code to be generated is fixed (invariant) at the input of the logic unit/prior to

performing logic operations. This again contributes to further reducing complexity of the logic unit and the corresponding step (while keeping the selectability of the type of code, where appropriate), because they need to be implemented for a single type of code only while the other types are generated by appropriately converting the index k.

Please delete the paragraph on page 10, lines 19-21.

Please amend the paragraph beginning on page 10, line 23 as follows:

~~Claims 3-6 and 14-17~~ Other embodiments of the present invention additionally provide advantageous implementations of the index conversion unit and the step of converting~~, respectively~~. They allow very low complexity and low delay realizations of OVSF-only ~~(claims 3, 4, 14, 15)~~, Hadamard-configurable, or OVSF/Walsh-configurable ~~(claims 6, 17) code generation apparati/methods~~ code generators and methods.

Please amend the paragraph beginning on page 11, line 9 as follows:

~~Claims 7, 8, and 18, 19~~ Other embodiments of the present invention additionally provide advantageous implementations of the logic unit and the step of performing logic operations, respectively. They allow very low complexity and low delay realizations of any kind of fixed-type ("hard-wired") or selectable-type code generation apparatus/method, because just binary AND and/or XOR operations are performed in order to calculate a code bit of the desired codeword.

Please amend the paragraph beginning on page 12, line 1 as follows:

According to a second aspect of the present invention, this object is achieved ~~by the~~ through the use of a parallel code generator ~~of claim 10~~. In particular, the object is achieved by the provision of (a) a total of p code generators ~~according to one of the claims 1 to 8 (i.e., not including a counter)~~ that do not include a counter, each for generating one of ~~said~~ the p orthogonal codes having a particular one of ~~said~~ the

indices, and (b) a counter for generating ~~said~~ the counter value i to be used by ~~said~~ the p code generators.

Please amend the paragraph beginning on page 12, line 12 as follows:

According to a third aspect of the present invention, this object is also achieved ~~by the~~ through the use of a parallel code generator ~~of claim 11~~. In particular, the object is achieved by the provision of p code generators ~~according to claim 9 (i.e., each including a counter)~~, each including a counter, and each for generating one of ~~said~~ the p orthogonal codes having a particular one of ~~said~~ the spreading factors and a particular one of ~~said~~ the indices.

Please amend the paragraph beginning on page 12, line 20 as follows:

The ~~features of claims 10 and 11~~ second and third aspects of the present invention advantageously allow ~~to concurrently generate~~ concurrent generation of several (p) codewords having different spreading factors SF and/or indices k ~~(optionally: and/or types)~~, the codewords optionally being of different types.

Please amend the paragraph beginning on page 12, line 24 as follows:

According to ~~claim 10~~ the second aspect, a single counter is provided in order to generate a counter value i to be used by all p code generators. This allows for a very low complexity implementation of the parallel code generator, which can be used in cases where the p desired codewords are to be generated synchronously, i.e. where the first code bits of the codewords are to be output at the same time.

Please amend the paragraph beginning on page 12, line 31 as follows:

According to ~~claim 11~~ the third aspect, each of the p code generators is provided with a dedicated counter. While increasing complexity when compared with the

implementation according to ~~claim 10~~ the second aspect, this allows for an asynchronous operation of the p code generators, where the first code bits of the codewords are not necessarily ouput at the same time.

Please delete the paragraph on page 13, lines 6-8.

Please amend the paragraph beginning on page 13, line 9 as follows:

As the skilled person will readily appreciate, variants other than those ~~according to claims 10 and 11~~ described above can easily be derived. For example, the counter could be split into several parts, wherein a first part could be used for all code generators (and therefore would be provided only once) while a second part of the counter could be dedicated to the p code generators (and therefore would be provided in each code generator).